Supplementary Material: Reclaiming Quantization Residual Knowledge

Róisín Luo (Jiaolin Luo)[1,2]
j.luo2@universityofgalway.ie
Alexandru Drimbarean[3]
James McDermott[1,2]
Colm O'Riordan[1,2]

[1] SFI Centre for Research Training in Artificial Intelligence
Dublin, D02 FX65, Ireland
[2] University of Galway
Galway, H91 TK33, Ireland
[3] Tobii Galway
Galway, H91 V0TX, Ireland

A Uniform quantization

We formally introduce uniform quantization, which refers to the integer representations of floating-point tensors by taking the quantization intervals uniformly [1, 2].

Suppose:

$$\lfloor \cdot \rceil : \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N} \mapsto \mathbb{Z}^{I_1 \times I_2 \times \cdots \times I_N} \tag{1}$$

is an element-wise rounding operator in tensor space such as $round(\cdot)$, $floor(\cdot)$ or $ceil(\cdot)$ in pytorch [4].

Quantization operator. Suppose:

$$[\![\cdot]\!]_n : \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N} \mapsto \mathbb{Z}^{I_1 \times I_2 \times \cdots \times I_N} \tag{2}$$

is a n-bit quantization operator which sends tensors from floating-point representations to n-bit integer representations.

De-quantization operator. We define the de-quantization operator as:

$$[\![\cdot]\!]_n^{-1} : \mathbb{Z}^{I_1 \times I_2 \times \cdots \times I_N} \mapsto \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}. \tag{3}$$

Let $x \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ be some floating-point tensor. Let $[\alpha, \beta] \subset \mathbb{R}$ be the quantization representation range (*i.e.* quantization clipping range) where $\alpha, \beta \in \mathbb{R}$. Choosing the clipping range constant $[\alpha, \beta]$ is often referred as 'calibration' [1]. Let $s \in \mathbb{R}$ be some scale constant determined by quantization bit-width and representation range $[\alpha, \beta]$:

$$s(n; \alpha; \beta) = \frac{\alpha - \beta}{2^n - 1}. \tag{4}$$

Suppose $z \in \mathbb{Z}$ denotes the integer representation zero point (*i.e.* quantization bias), the quantization operator $[\![\cdot]\!]_n$ can be formulated as:

$$x^* = [\![x]\!]_n \stackrel{def}{=} \lfloor \frac{x}{s(n;\alpha;\beta)} \rceil - z \tag{5}$$

which represents x into the range:

$$[\lfloor \frac{\alpha}{s(n;\alpha,\beta)} \rceil - z, \lfloor \frac{\beta}{s(n;\alpha,\beta)} \rceil - z] \subset \mathbb{Z}. \tag{6}$$

The choices of z determine two schemes for uniform quantization: (1) Symmetric quantization scheme and (2) asymmetric quantization scheme (*i.e.* affine quantization) [1, 3, 6].

For example, the scheme $z := \frac{\alpha}{s(n;\alpha,\beta)}$ is dubbed as 'asymmetric quantization scheme' or 'affine quantization scheme' where the floating-point representation zero-point is mapped to the bias z. The scheme $z := 0$ is dubbed as 'symmetric quantization scheme' where the floating-point representation zero-point is mapped to zero.

Accordingly, the de-quantization operator $[\![\cdot]\!]_n^{-1}$ can be formulated as:

$$x = [\![x^*]\!]_n^{-1} \stackrel{def}{=} s(n;\alpha,\beta) \cdot (x^* + z). \tag{7}$$

B Mode-*n* tensor product

Let $Z \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ be a N-order tensor where I_i denotes the i-th dimension size. A ‘fiber’ refers to the vector created by fixing $N-1$ dimensions. A ‘slice’ refers to the matrix created by fixing $N-2$ dimensions.

The mode-n matricization of tensor Z is also known as tensor ‘unfolding’. For example, a tensor with shape $8 \times 16 \times 3 \times 3$ can be unfolded into a matrix with shape 8×144. The mode-n matricization of tensor Z is denoted as $Z_{(n)}$ by arranging the mode-n fibres of Z as columns. The mode-n product is known as tensor-matrix product. The mode-n product of tensor Z and matrix $Y \in \mathbb{R}^{J \times I_n}$ is defined as $Z \times_n Y \overset{def}{=} Y Z_{(n)}$.

Accordingly, we also define the inverse mode-n tensorization (*i.e*. ‘folding’) as the inverse operation and denote it as $Z_{[n, J_1 \times J_2 \times \cdots \times J_K]}$ where $J_1 \times J_2 \times \cdots \times J_K$ denotes the folding dimensions of the unfolded dimensions in $Z_{(n)}$. Readers can refer to the literature [5] for details.

C Proof: Residual convolutional representation

We aim to prove:

$$W \circledast x = [\![W]\!]_n \circledast x + \underbrace{B \circledast A}_{\text{residual operator}} \circledast x \tag{8}$$

as stated in Theorem 1.

However, the convolutional operations are not matrix multiplications. We can not directly use the results such as singular value decomposition (SVD). Strictly proving the Theorem 1 demands some efforts. We use the knowledge from tensor algebra to show that Theorem 1 holds.

Definition 1 (Unfolding operator). *Let:*

$$\mathbb{T}(k_1 \times k_2, s_1 \times s_2) : \mathbb{R}^{n \times w \times h} \mapsto \mathbb{R}^{n \cdot k_1 \cdot k_2 \times w' \times h'} \tag{9}$$

be the 'unfolding' operation which arranges some input with shape '$n \times w \times h$' to shape '$n \cdot k_1 \cdot k_2 \times w' \times h'$' for convolution operation with respect to kernel size $k_1 \times k_2$ and stride size $s_1 \times s_2$. In particular, if the stride size is 1×1, we simplify the notation as:

$$\mathbb{T}(k_1 \times k_2). \tag{10}$$

Suppose $x \in \mathbb{R}^{n \times w \times h}$. For example, the operator with stride $s_1 \times s_2$ and no padding is defined by:

$$\mathbb{T}(k_1 \times k_2, s_1 \times s_2)(x)(c, u, v) := x(c \mod (n \cdot k_1 \cdot k_2), \lfloor \frac{u}{s_1} \rfloor, \lfloor \frac{v}{s_2} \rfloor) \tag{11}$$

where c, u and v are indices. Clearly:

$$\mathbb{T}(1 \times 1)(x) \equiv x \tag{12}$$

holds true as a particular case. Readers can refer to the implementation of the operation unfold in pytorch.

Lemma 1 (Tensor mode-n product factorization). *Let $Z \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ be some tensor. Let $A \in \mathbb{R}^{R \times I_n}$ and $B \in \mathbb{R}^{J \times R}$ be matrices. Below identity holds:*

$$Z \times_n A \times_n B \equiv Z \times_n (BA). \tag{13}$$

Readers can refer to literature [5] for the proof.

Lemma 2 (Convolution mode-n representation). *Let $W \in \mathbb{R}^{m \times n \times k_1 \times k_1}$ be the weights of some '$k_1 \times k_1$' 2D convolutional operator where m denotes output channels and n denotes input channels. Let $x \in \mathbb{R}^{n \times w \times h}$ be some input with size $w \times h$ and channels n. According to the definition of 2D convolution, the convolution $W \circledast x$ can be represented as:*

$$W \circledast x = \underbrace{\mathbb{T}(k_1 \times k_2)(x)}_{n \cdot k_1 \cdot k_2 \times w' \times h'} \times_1 W_{(1)}. \tag{14}$$

Theorem 3 (Convolution factorization). *Let $Z \in \mathbb{R}^{m\times n\times k_1\times k_2}$ be the weights of some 2D convolutional operator. Suppose:*

$$Z_{(1)} = B_{(1)}A_{(1)} \in \mathbb{R}^{m\times n\cdot k_1\cdot k_2} \tag{15}$$

where $A \in \mathbb{R}^{d\times n\times k_1\times k_2}$ and $B \in \mathbb{R}^{m\times d\times 1\times 1}$ are the weights of two convolutional operators with kernel sizes '$k_1 \times k_1$' and '1×1' respectively. Let $x \in \mathbb{R}^{n\times w\times h}$ be some input. Using Lemma 2 and Lemma 1:

$$\begin{aligned}
Z \circledast x &= \mathbb{T}(k_1 \times k_2)(x) \times_1 Z_{(1)} && (16)\\
&= \mathbb{T}(k_1 \times k_2)(x) \times_1 (B_{(1)}A_{(1)}) && (17)\\
&= \mathbb{T}(k_1 \times k_2)(x) \times_1 A_{(1)} \times_1 B_{(1)} && (18)\\
&= (A \circledast x) \times_1 B_{(1)} && (19)\\
&= \mathbb{T}(1 \times 1)(A \circledast x)B_{(1)} && (20)\\
&= B \circledast A \circledast x. && (21)
\end{aligned}$$

Corollary 4 (Convolutional singular value decomposition). *Suppose:*

$$Z_{(1)} = USV^T = US^{\frac{1}{2}}(S^{\frac{1}{2}}V)^T \tag{22}$$

where:

$$S^{\frac{1}{2}} \odot S^{\frac{1}{2}} = S. \tag{23}$$

Set:

$$B_{(1)} := US^{\frac{1}{2}} \tag{24}$$

and:

$$A_{(1)} := (S^{\frac{1}{2}}V)^T. \tag{25}$$

Using Theorem 3:

$$Z \circledast x = (US^{\frac{1}{2}})_{[1,d\times 1\times 1]} \circledast (S^{\frac{1}{2}}V)^T_{[1,n\times k_1\times k_2]} \circledast x. \tag{26}$$

Proof. We now show that the Theorem 1 strictly holds by using Theorem 3 and Corollary 4. Suppose $W \in \mathbb{R}^{m\times n\times k_1\times k_2}$ be the weights of some convolutional operator. Set:

$$Z := \Delta[\![W]\!]_n \in \mathbb{R}^{m\times n\times k_1\times k_2}. \tag{27}$$

Convolutional operators are linear operators. The convolution quantization residual representation is:

$$\begin{aligned}
W \circledast x &= [\![W]\!]_n \circledast x + \Delta[\![W]\!]_n \circledast x && (28)\\
&= [\![W]\!]_n \circledast x + (US^{\frac{1}{2}})_{[1,d\times 1\times 1]} \circledast (S^{\frac{1}{2}}V)^T_{[1,n\times k_1\times k_2]} \circledast x. && (29)
\end{aligned}$$

There is nothing to do. Theorem 1 holds as demonstrated. □

D Rank normalization coefficients

Suppose a model with L convolutional filters. Suppose the l-th layer has parameter size Θ_l. Suppose the adaptation of the i-th layer has parameters Ξ_i. The maximum parameter size of the l-th adapter is Θ_l.

The overall size of the adapters is given by:

$$\frac{r_l}{R_l} \cdot \Theta_l. \tag{30}$$

Normalizing with respect to the overall model size:

$$\sum_{i=1}^{L} \Theta_i. \tag{31}$$

Thus, the running budget is:

$$\frac{r_l}{R_l} \cdot \Theta_l \cdot \frac{1}{\sum_{i=1}^{L} \Theta_i} \leq b. \tag{32}$$

Set:

$$\omega_l = \frac{1}{R_l} \cdot \frac{\Theta_l}{\sum_{i=1}^{L} \Theta_i} \tag{33}$$

which is referred as l-th layer *rank normalization coefficient*.

E Equivalent quantization bit-width

Suppose the l-th layer parameter size Θ_l. The the l-th layer adapter size is:

$$\frac{r_l}{R_l} \cdot \Theta_l. \tag{34}$$

The equivalent quantization bit-width ξ is:

$$\frac{\xi}{32} = \frac{\frac{n}{32} \cdot \sum_{i=1}^{L} \Theta_i + \frac{m}{32} \cdot \sum_{i=1}^{L} \frac{r_l}{R_l} \cdot \Theta_i}{\sum_{i=1}^{L} \Theta_i}. \tag{35}$$

Simplifying the Equation (35):

$$\xi = n + m \cdot b. \tag{36}$$

F Full solutions

We provide full solutions for all experimental models.

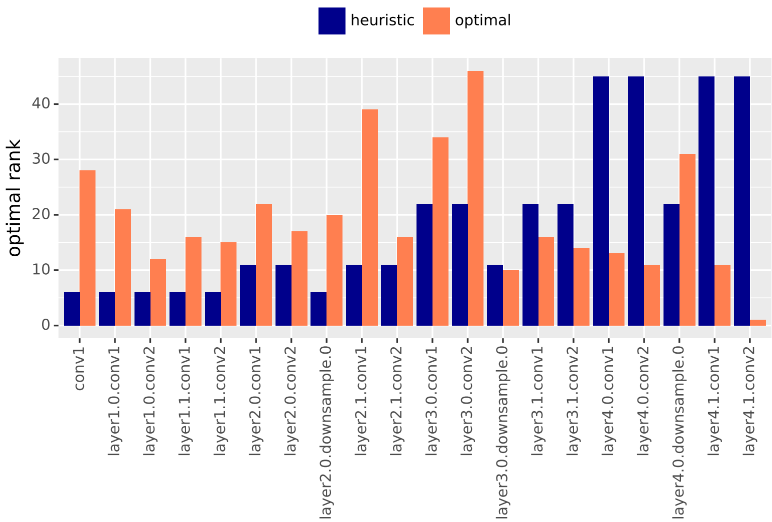

Figure 1: Solution for *resnet18*.

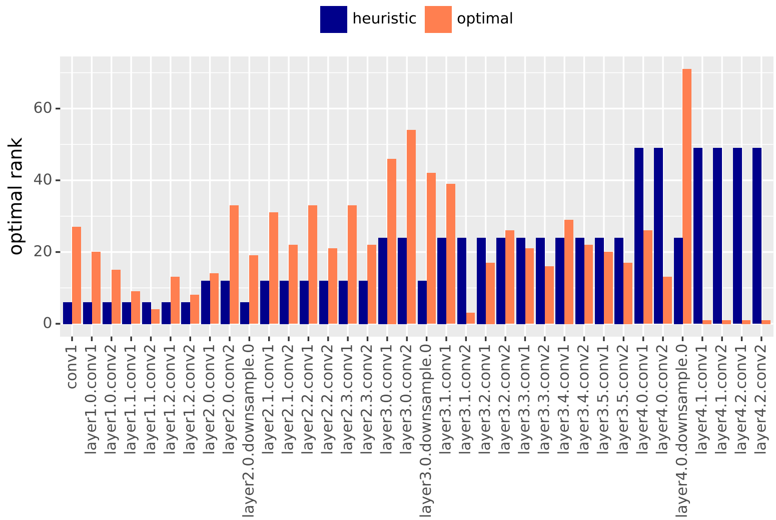

Figure 2: Solution for *resnet34*.

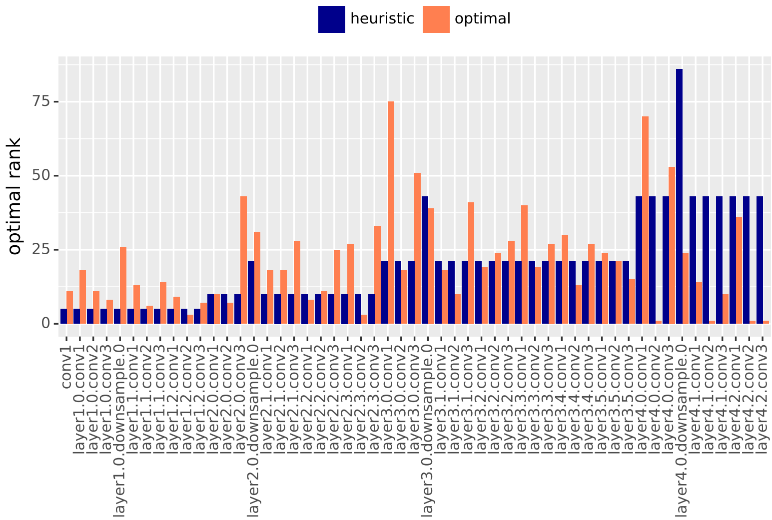

Figure 3: Solution for *resnet50*.

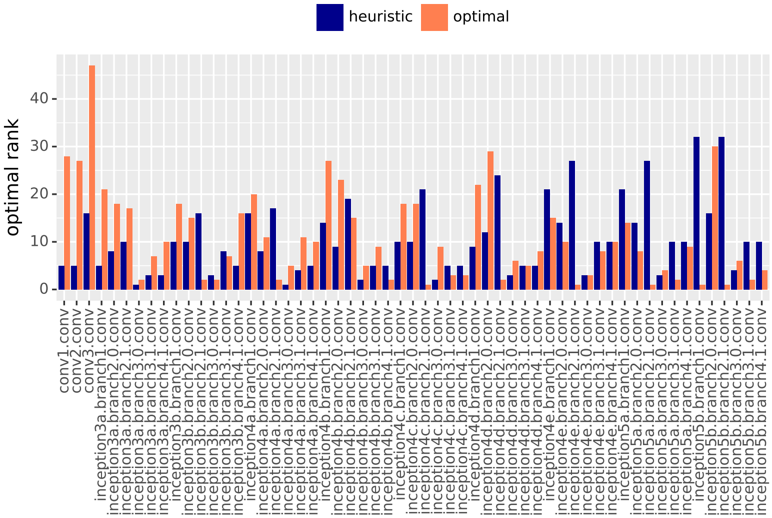

Figure 4: Solution for *inception*.

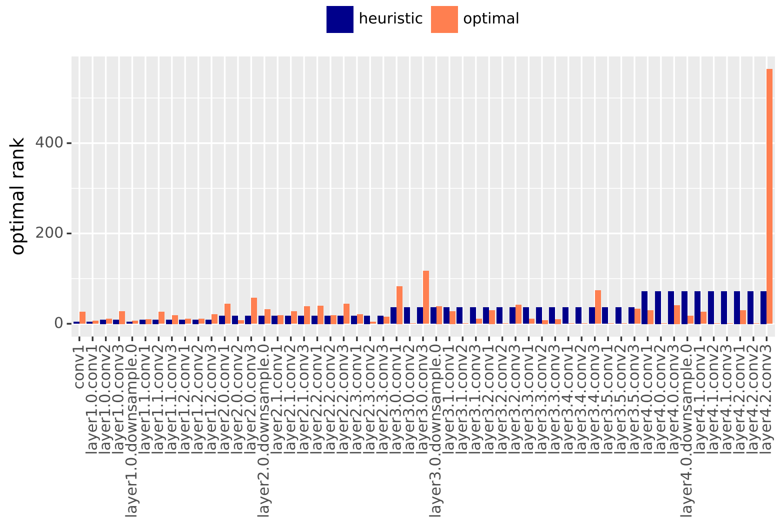

Figure 5: Solution for *wide_resnet50_2*.

References

[1] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*, pages 291–326. Chapman and Hall/CRC, 2022.

[2] Yunhui Guo. A survey on methods and theories of quantized neural networks. *arXiv preprint arXiv:1808.04752*, 2018.

[3] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713, 2018.

[4] Nikhil Ketkar, Jojo Moolayil, Nikhil Ketkar, and Jojo Moolayil. Introduction to pytorch. *Deep Learning with Python: Learn Best Practices of Deep Learning Models with PyTorch*, pages 27–91, 2021.

[5] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[6] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018.